13. ročník
Žilina 1. – 3. 7. 2025
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.fri.uniza.sk
OSS Conf

OSSConf 2025: 1–8

Míchání rastrové a vektorové grafiky v typografii

PAVEL STŘÍŽ (CZ)

Abstrakt. Článek představuje některé styčné body kombinování vektorové a rastrové grafiky na úrovni přípravy mraků slov.

Klíčová slova. Mrak slov, TEX, Python, FontForge, Inkscape, GIMP, kvadrantové stromy.

COMBINING RASTER AND VECTOR GRAPHICS IN TYPOGRAPHY

Abstract. The paper introduces basic concepts in typography of combining vector and raster graphics at a preparation of wordclouds level.

Keywords. Wordclouds, Tag clouds, TEX, Python, FontForge, Inkscape, GIMP, quadtrees.

1. O problému

Kolem roku 2007 jsem sázel popisky na CD a DVD a chtěl jsem vysázet drobnost, a to, že rok bude vysázen tak, aby se cifry dotýkaly. Měla to být má tečka za konferenční řadou TEXperience.

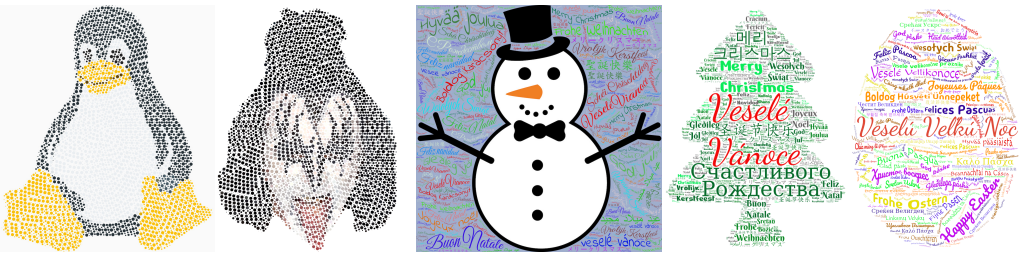
2007! → 2007!

V TEXu získáme výsledek ručně za pár minut po několika iteracích, třeba takto:

```
{\bfseries\huge2\kern-2.4pt0\kern-1.9pt0\kern-2.4pt7\kern-2.4pt!}
```

Pokud se nad tím zamyslíme hlouběji a chceme-li takovou sazbu zautomatizovat, dostaneme se na úroveň křivek glyfů. K těm se dá dostat například přes METAPOST či program FontForge. Zjistit dotyk křivek je ale výpočetně náročná úloha. A to mluvíme o dvou křivkách, žádných složitých obrazcích. Začal jsem s tím experimentovat. Bohužel na zjištění toho, jestli je bod uvnitř či vně jisté křivky jsem použil nevhodný postup a mé cvičné pětihvězdičky jsem nebyl schopen vysázet na obálku tehdy vznikající knihy.

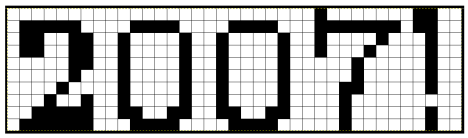
Matematik Honza Šustek mi za jeden večer poslal řešení v METAPOSTu na poskládání glyfů. Tam však byla otázka, jak přistupovat k libovolným obrázkům s blíže nespecifikovaným počtem křivek. Problém jsem odložil v duchu, že je nad mé síly. Po letech, v roce 2015, mi to stále vrtalo hlavou a vznesl jsem dotaz na TEX.SE, `https://tex.stackexchange.com/questions/180510/how-to-get-intersection-points-of-two-glyphs`, ale neposunul jsem se. Zde je několik vzorků ze světa wordcloudů, to byla jistá inspirace, jak by vzorky mohly vypadat. Ukázky jsou od Aleše Kozubíka za pomoci `https://www.wordclouds.com/` a `https://wordart.com/`.



2. D3.js

S příchodem tohoto nástroje jsem si všiml častého užití rekurzívních datových struktur quadtree (kvadrantový strom) a octree (oktálový strom), viz například Poisson Disc Sampling, `https://www.jasondavies.com/poisson-disc/`. Kdybych to měl amatérsky přiblížit, tak je to rozšíření metody půlení intervalu v 1D pro 2D a 3D. Použití je masivní hlavně u vícerozměrných dat (typu mraků dat v geoinformatice či herním průmyslu) a rastrových obrázků (arkádové hry). Tehdy začal vznikat nápad, že by se, teoreticky řečeno, dal vektorový výstup sazby vzít, převést do rastrového obrázku, nějak s tím pracovat, a výsledky použít znovu do sazby, ale pro původní vektorové podklady. Teoreticky by mělo docházet k chybě maximálně jednoho pixelu, s možností si rastrový obrázek zjemnit, kdyby bylo potřeba. Pozn. o bojích s písmy více Sebastian Lague, `https://www.youtube.com/watch?v=SO83KQuuZvg`, o nových písmech hovoří Erik Demaine, `https://www.youtube.com/watch?v=zZcJq0T_FP8`.

Zde je školní ukázka. Pokud provedeme posun znaků o 2 (první nula), 4 (druhá nula), 6 (sedmička) a 7 pixelů (vykřičník) dostaneme se na jejich dotyky. V každém případě vidíme, že počet pixelů neodpovídá posunům v první ukázce (jednotky pt). Pokud si velikost původního obrázku přepočteme na pixely, můžeme provést posun i v obrázku vektorovém. Pokud by chyba pixelu byla viditelná, zjemníme si rastrový obrázek, tím dojde k přesnějšímu výsledku.



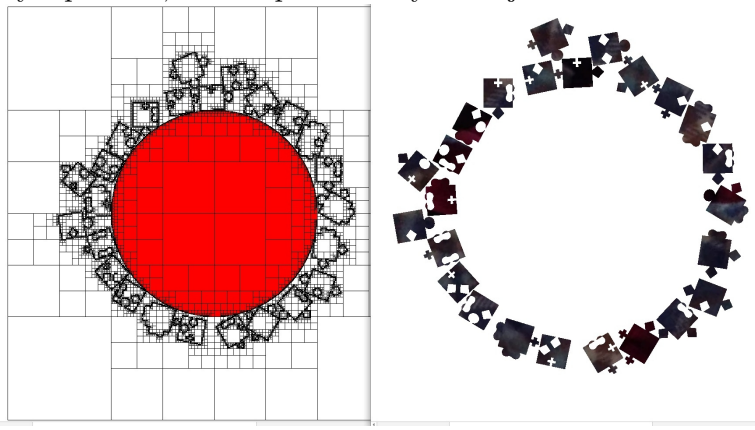
Jistý nápad to sice byl, ale rekurzívní datové struktury nebyly můj denní chleba, tak jsem se do testů nepouštěl. Jistou práci s XML, SVG či JSON formáty jsem měl sice za sebou, ale tohle bylo ještě o kus dále. Bez nich to však prakticky u rastrových obrázků nejde skrz nutný výpočetní čas.

3. Puzzle

V mezičase jsem řešil kombinatorické úlohy. Jedna z nich byla spočítat, kolik jedinečných dílků (i s ohledem na rotaci) lze získat u puzzle. Matematik by na to šel

přes Burnsideův teorém, programátor přes zjištění kolik dílků je jedinečných/rotačně invariantních z celkového počtu. Obě cesty mi zafungovaly a za odměnu jsem si chtěl puzzle dílky nechat vykreslit. Lze to udělat náhodně, ale dílky by šly přes sebe a na kontrolu by to nebylo vhodné. Tak jsem použil vysázení dílků do mřížky. Tím jsem se pochlubil Alešovi Kozubíkovi do Žiliny. Odpověď byla jasná: není to náhodné, není to hezké. Tak jsem si říkal, že bych mohl zkusit to vysázet do spirály, přesně tak, jak je to v pozadí u wordcloudů. O těch jsem věděl, že právě užívají kvadrantové stromy.

Pustil jsem se do prvních pokusů. Málem jsem spadl ze židle, když jsem viděl svůj první zdařilý pokus: puzzle dílky naskládané jakoby mimo kruh (dále se to nazývá jako mustr). Zde je onen obrázek. V levé části je vidět vizualizace konečného stromu, v pravé části přechod zpět do vektorového světa sazby. A nejen to, kdyby bylo potřeba, tak lze puzzle dílky sázet jako rastrové obrázky.



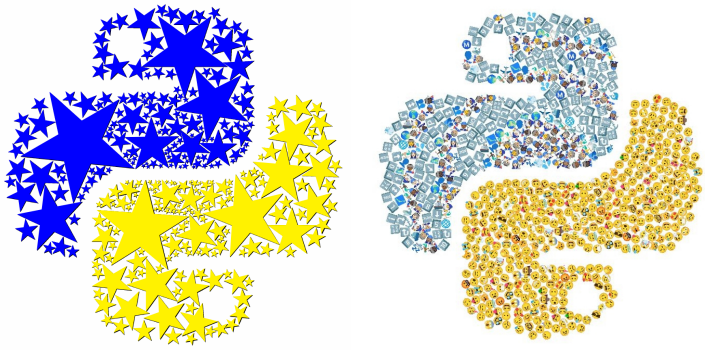
Abych zahladil efekt umisťování na spirálu, tak jsem si dílky náhodně zmenšoval. Dokonce lze ze spirály přejít na jiný typ (po řádcích, po sloupcích, více menších spirál, či dle libovůle). Ale za základ testů jsem vzal spirálu.

Kvadrantový strom se mi buduje tak, že zkoumám, jestli jsou v kvadrantu pixely bílé či černé, přiřadím kvadrantu nulu či jedničku. Pokud je však kvadrant směsí bílých a černých, rozčtvrtí se oblast a zkoumá rekurzívně hlouběji. Tímto způsobem si lze libovolný obrázek rozebrat na soustavu černých obdélníků, u kterých je jednoduché hledat průsečíky s jinými obdélníky. To je výpočetní jádro. Postupně z fronty obrázků zjišťujeme, jestli je lze či nelze umístit na spirále. Upraví se mustr a poté se ve frontě přejde na další obrázek.

4. Ukázky z vlastní zahrádky

Když umíme obrázky takto skládat do mustru. Už to není o ničem jiném než o přípravě mustru a vhodné přípravě podkladů. Naprogramoval jsem si fíčury, že si mohu definovat oblasti barev a také oblasti, kde se mohou objevovat jen chtěné symboly. Toho jsem využil například u loga Pythonu, kdy jsem upřesnil barvy

(obarvené hvězdičky, ukázka vlevo) či typ (vybrané žluté a modré emoji, ukázka vpravo). Přidal jsem přepínač, zda-li může či nemůže podklad do jiné oblasti.



Abych se dokázal zpátky vrátit do původní sazby používám TEXový příkaz `\typeout`. Pak nemusím do konečné sazby užívat průběžné rastrové (např. PNG) či vektorové obrázky (např. PDF). Jedná se o řádovou úsporu velikosti souboru. Do článku se mi mnoho ukázek nevleze, ale příchozí účastníci na OSSConf si budou moci v Žilině prohlédnout výstavu. Některé vzorky budou vlastní a některé inspirativní wordcloudy budou od jiných autorů.

5. První zákazníci

Ačkoliv se mi zatím nepodařilo napojování kvadrantových stromů, dělám to zatím úpravou rastrového obrázku a novým generováním stromu, je program relativně pomalý. I tak jsem dal organizátorům OSSConf vědět, že je to již použitelné. Mezi první „zákazníky“ lze počítat Aleše Kozubíka a Mila Ofúkaného.

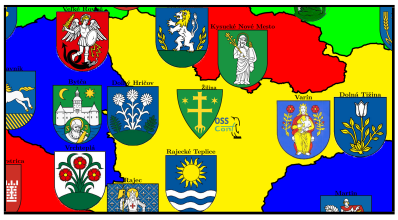
Pro Aleše vzniká logo konference, obrázky na jednotlivé sekce a experimenty s grafikou hudební skupiny KISS.

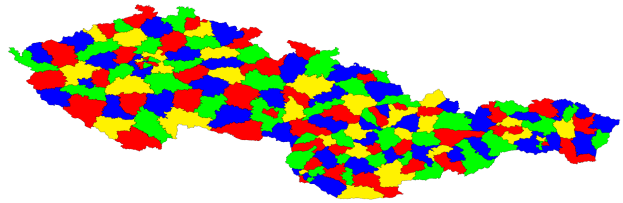
Milo Ofúkaný má obrovský záběr a řadu nápadů. Zkusili jsme doladit loga pro jeho sekci (Open GIS & Open Data).

6. Mapa ČR a SR a teorém 4 barev

Zesnulý TEXpert Karel Horák se těšil na mou knihu erbů s popisky. Před mnoha lety se mi podařilo získat data za Českou republiku (systém REKOS), ale nikdy jsem neměl dost dat za Slovensko, tak jsem se do toho nepustil. Stále jsem erby měl v hlavě, a říkal jsem si, že zkusím alespoň mapu. Za pomoci dat Michala Páleníka, `https://www.oma.sk`, mapových podkladů z OpenStreetMap a údajů z Wikidat jsem začal mapu skládat.

Aleš Kozubík nadhodil nápad na problém 4 barev a jestli jej nějak nezapracovat. Připomínky posílal Rudolf Blaško, matematický model nezávisle testoval Štefan Peško. Já se naučil model ILP generovat pomocí Pythonu jako .pi soubor pro program Picat, `https://picat-lang.org/`. Zde je ukázka z pracovní verze z okolí Žiliny a jak vypadá vykreslení okresů, aby sousedi neměli stejnou barvu.





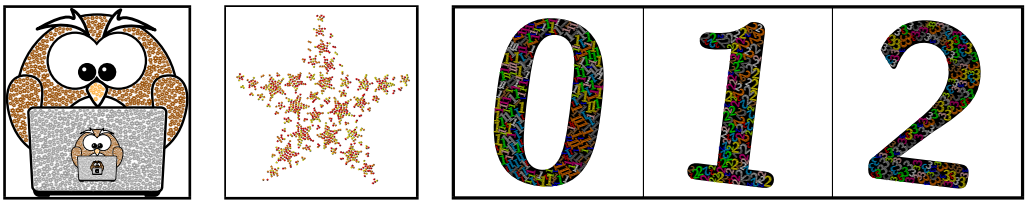
7. Rekurzívně

Vedle matematického modelování se snažím řešit úlohy rekurzí. Ne vždy je to technicky a časově možné, ale držím se tímto v programátorské formě. Proto jsem i u wordcloudů zkoumal možnost využití.

První možnost je jednoduchá a intuitivní. Vezmeme si hotový obrázek, v ukázce tučňák s notebookem, a v sazební postprodukci si vnořování připravíme jako vrstvení obrázků s jejich postupným zmenšováním.

Druhá možnost, která mě napadla, byla, že vezmeme nově vzniklý černobílý mustr a dále jej použijeme. Nemám to dostatečně zautomatizované, zde je ukázka hvězdy z hvězdiček z hvězdiček. Varianta to není ideální, protože si musíme hlídat velikost rastrového mustru. Čím se jde do větší hloubky, tím ztrácíme přesnost. A na úrovni jednotek pixelů už nemá smysl pracovat. Proto jsem přemýšlel, jak na to jinak.

Třetí ukázka je zautomatizovaná verze, kdy si připravujeme podklady jakoby odzadu a postupně z nich sazbu vystavíme. V této ukázce je nula vyplněna jedničkami, ty jedničky jsou vyplněny dvojkami atd. Každá barva reprezentuje jeden typ, celkem bylo cca tucet barev. V ukázce je vždy první typ. Kompletní sazbu všech symbolů na jednu stranu nelze zrealizovat, protože PDF by dosáhlo neuvěřitelné velikosti. Můžeme však vnořování naznačit. Pro mne to byla fascinující úloha, u které jsem nevěřil, že s ní pohnu.



8. Několik tipů na nástroje

Ze starších nástrojů mohu zmínit METAPOST a pstoedit. Bohužel METAPOST má omezení na práci s velkými čísly a je potřeba škálovat. Také pravidlo eofill je potřeba si ohnout nebo užít balíček luamplib, neb to Hans Hagen vyřešil.

Nástroj pstoedit umí vzít PostScriptový soubor a vygenerovat zdrojový kód pro METAPOST. Dokonce umí SVG, autor psal, že některé části ještě veřejně nezveřejnil, ale že to plánuje. Bohužel se mi stalo, že občas konverze nedopadla dobře. Vypadá to tedy, že zpracovávat přímo SVG se jeví jako budoucnost. Můj pracovní cíl totiž je, že bych rád objekty vyřezával nikoliv používal barvu bílou či pozadí obrázku.

GIMP asi netřeba představovat. Programy ImageMagick a GraphicsMagick pomohly s následujícím: nechat levý a pravý dolní roh zaplnit průhledností. Užitečný je i přepínač `trim`, který promaže ochrannou zónu obrázku:

```
read w h < <(gm identify -format "%w %h" vstup.png)
let w=$w-1; let h=$h-1
gm convert vstup.png -matte -fill transparent temp.png
convert temp.png -fuzz 40% -fill none -draw "matte $w,$h floodfill" \
  -draw "matte 0,$h floodfill" -trim vysledek.png
```

První řádek zjistí velikost vstupního obrázku. Druhý řádek poníží zjištěnou velikost o jeden pixel, kvůli číslování pixelů od nuly. Třetí řádek zajistí, že obrázek bude mít průhledné pozadí. Poslední řádek využije spodních rohů a pokusí se zajistit průhlednost kolem erbu.

Program Inkscape asi taky netřeba blíže představovat. Umí však i práci z příkazového řádku. První řádek vezme vstupní SVG a vygeneruje PDF. Co jsem u příprav zjistil je, že umí i zpracovat kroky, jako kdybych je klikal myší. Na druhém řádku je ukázka, že se SVG prvně zmenší na velikost obrázku a pak vygeneruje PNG.

```
inkscape --export-filename=out.pdf in.svg
inkscape --actions=page-fit-to-selection --export-filename=out.png in.svg
```

FontForge je program na práci s písmy. Má své rozhraní, soubory s příponou .pe. Zde je ukázka, kterou lze spustit přes `fontforge -script malskript.pe`. Otevře se soubor s písmy, upraví se znak A a výsledek se uloží do nového souboru.

```
Open("MalPokus.otf");
Select("A");
ExpandStroke(10);
Simplify();
Save("MalPokusOut.otf");
```

Autor připravil i rozhraní pro Python. Zde je ukázka `svg-na-znak.py`, kdy lze do písma na určitou pozici uložit SVG obrázek.

```
import fontforge
maloutput=fontforge.font()
glyph=maloutput.createMappedChar("A")
glyph.importOutlines("owl_notebook.svg")
maloutput.fontname="MalPokus"
maloutput.save("MalPokus.otf")
```

Tímto způsobem jsem si připravil masku cifer z písma `https://www.wfonts.com/font/the-kiss-font`. V TEXu jsem si znaky navrstvil. Je použit formát LuaLaTeX. O upravené písmo si případně napište, zatím nikde není.

```
\documentclass{standalone}
\usepackage{xcolor}
\usepackage{fontspec}
\setmainfont[Path=,Extension=.otf]{TheKISSFont}
\begin{document}
\makebox[0pt][l]{\color{red}\char"80}\char"00BA
\makebox[0pt][l]{\color{green}\char"80}\char"03A9
\makebox[0pt][l]{\color{blue}\char"80}\char"00E6
\makebox[0pt][l]{\color{red}\char"80}\char"00F8
\makebox[0pt][l]{\color{green}\char"80}\char"00BF
\makebox[0pt][l]{\color{blue}\char"80}\char"00A1
\makebox[0pt][l]{\color{red}\char"80}\char"00AC
\makebox[0pt][l]{\color{green}\char"80}\char"221A
\makebox[0pt][l]{\color{blue}\char"80}\char"0192
\makebox[0pt][l]{\color{red}\char"80}\char"2248
\end{document}
```

Takto vypadal výsledek. Zaujalo mě, že cifry 6 a 9 jsou rozdílné, to byl dobrý nápad. U této ukázky je vidět, že by nám obrys černých linek nestačil.

Barevná písma jsou stále nová a zkoumaná oblast. V TEXovém světě jsem to poprvé zahlédl u balíčků chessboard a xskak. V posledních letech Sam Carterová vyvíjí zvířátka a postavy, technicky je to přes Ti*k*Z, tedy velikost je o něco větší než kdyby to bylo přes znaky písma. Zaujatý čtenář nechť prozkoumá balíčky tikzlings, tikzmarmots a tikzducks.

9. Budoucnost a plány

V době psaní tohoto článku jsem stále ve fázi bádání a experimentování. Vize však je, že by mohl vzniknout nástroj s několika fíčurami a měl by skončit zdokumentovaný na `ctan.org`. Původní nápad byl, že průvodce dokumentací bude bobr Erl a jeho proslavené stavby ze dřeva. Postupem času mě napadlo, že natolik míchám rastrový a vektorový svět, že se z toho stává blázinec. Tak jsem se přiklonil ke kočkopsovi (angl. catdog), tedy k vnitřnímu boji, resp. spolupráci, těchto dvou světů.

Mezi otevřené a řešené partie patří:

- Vzít znak a rozpůlit jej. Lze to řešit pomocí Voronoi diagramu. Výsledky povzbudivé nejsou. Vezmeme-li znak A, tak užití dvou barev je zvláštní, spíš se to tváří na tři barvy. Pokud se to podaří, bude to atypická záležitost.
- Podobně mám otevřený problém přípravy písma, tzv. ementálu. Tedy díry v písmu budou skutečné díry, nikoliv překreslení bílou barvou či barvou pozadí obrázku. Jistá vize by byla, ale chce to čas.
- Podařilo se mi půlit znaky za pomocí rastrového obrázku, viz ukázka. Chtělo by to však přejít na automatizované vyřezávání vektorové. Například podle barvy křivek.

V každém případě se mohu přiznat, že tohle byl můj letitý otevřený projekt, ať dělám na libovolné pasáži, učím se něco nového. Jestli se skutečně podaří jej ještě zrychlit a dostat do podoby použitelného nástroje pro veřejnost, to se ještě uvidí, byla by to však třešnička na dortu.

Kontaktní adresa

Ing. Pavel Stříž, Ph.D., Nakladatelství Martin Stříž, U Škol 940, Bučovice, okres Vyškov, 685 01, Česká republika,
E-mailová adresa: `pavel@striz.cz`